Building a provenance-based intrusion detection system

Thomas Pasquier

University of Bristol

Talk loosely based on following publications

- Han et al. **“UNICORN: Revisiting Host-Based Intrusion Detection in the Age of Data Provenance”**, NDSS 2020
- Pasquier et al. **“Runtime Analysis of Whole-System Provenance”**, ACM CCS 2018
- Han et al. **“Provenance-based Intrusion Detection: Opportunities and Challenges”**, USENIX TaPP 2018
- Pasquier et al. **“Practical Whole-System Provenance Capture”**, ACM SoCC 2017

Partners

Institutions (core)

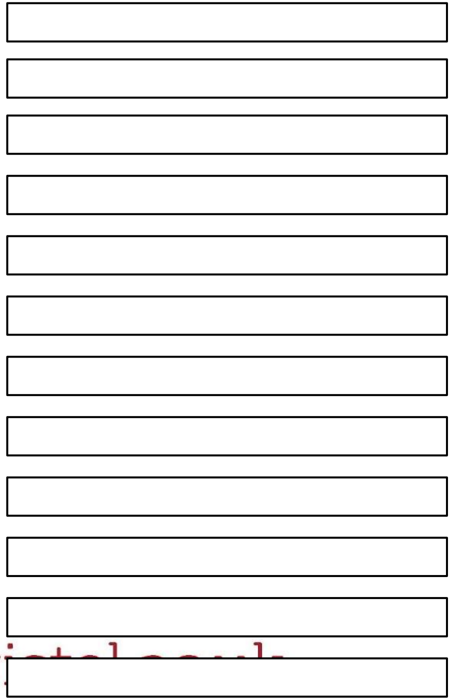
- University of Cambridge (UK)
- Harvard University (US)
- University of British Columbia (Canada)

Funding

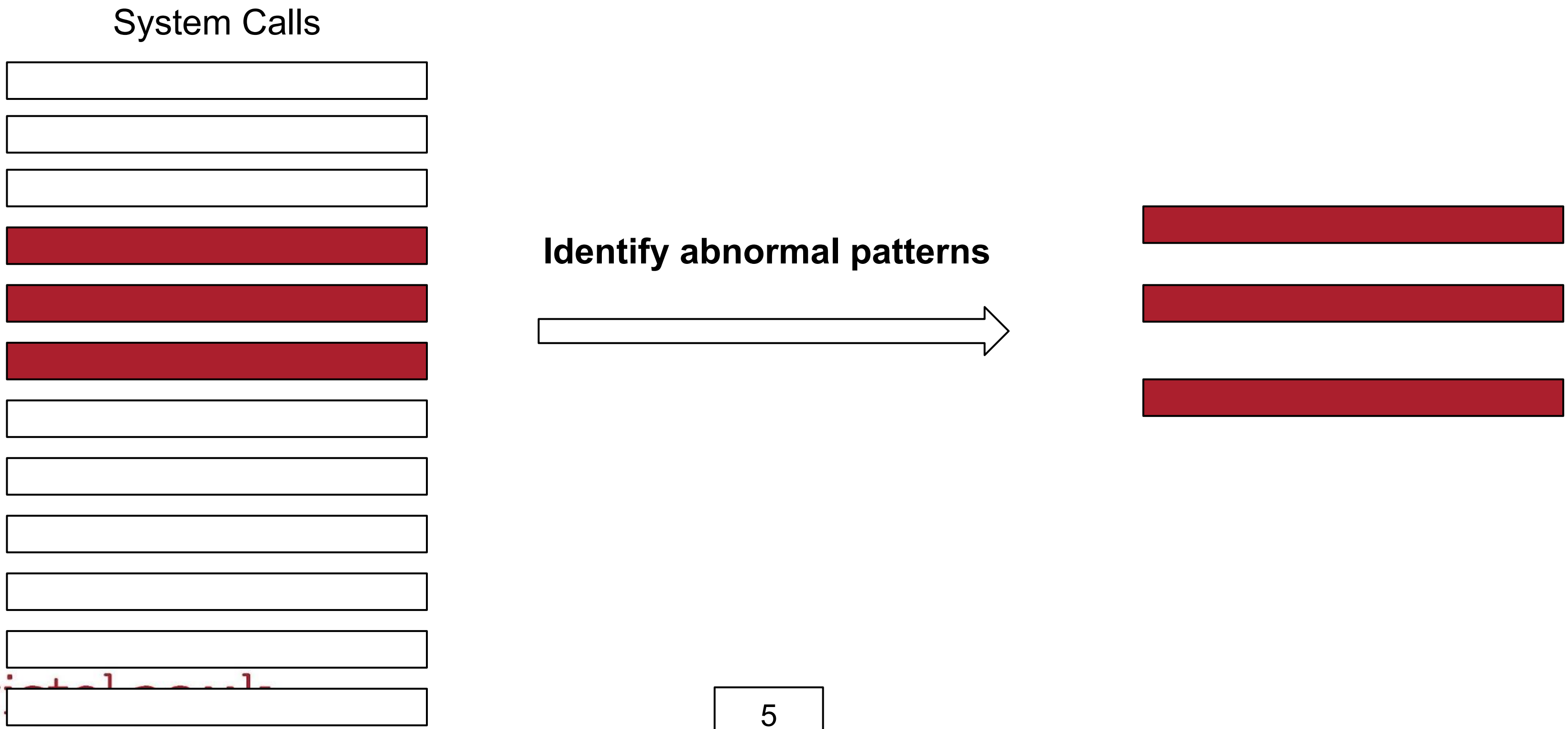
- EPSRC
- NSF
- DARPA
- Microsoft Cloud Computing Research Centre

System call based intrusion detection

System Calls

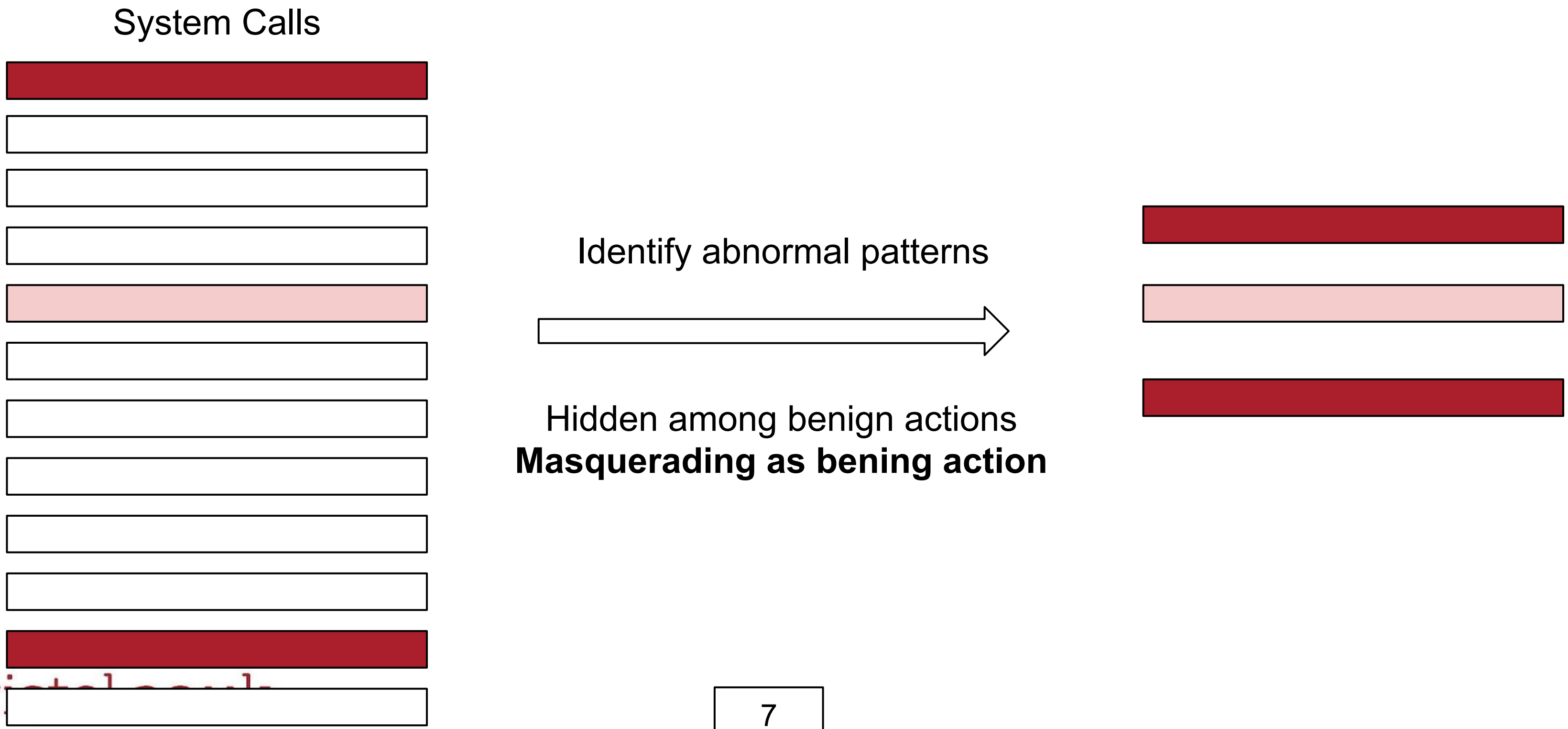


System call based intrusion detection

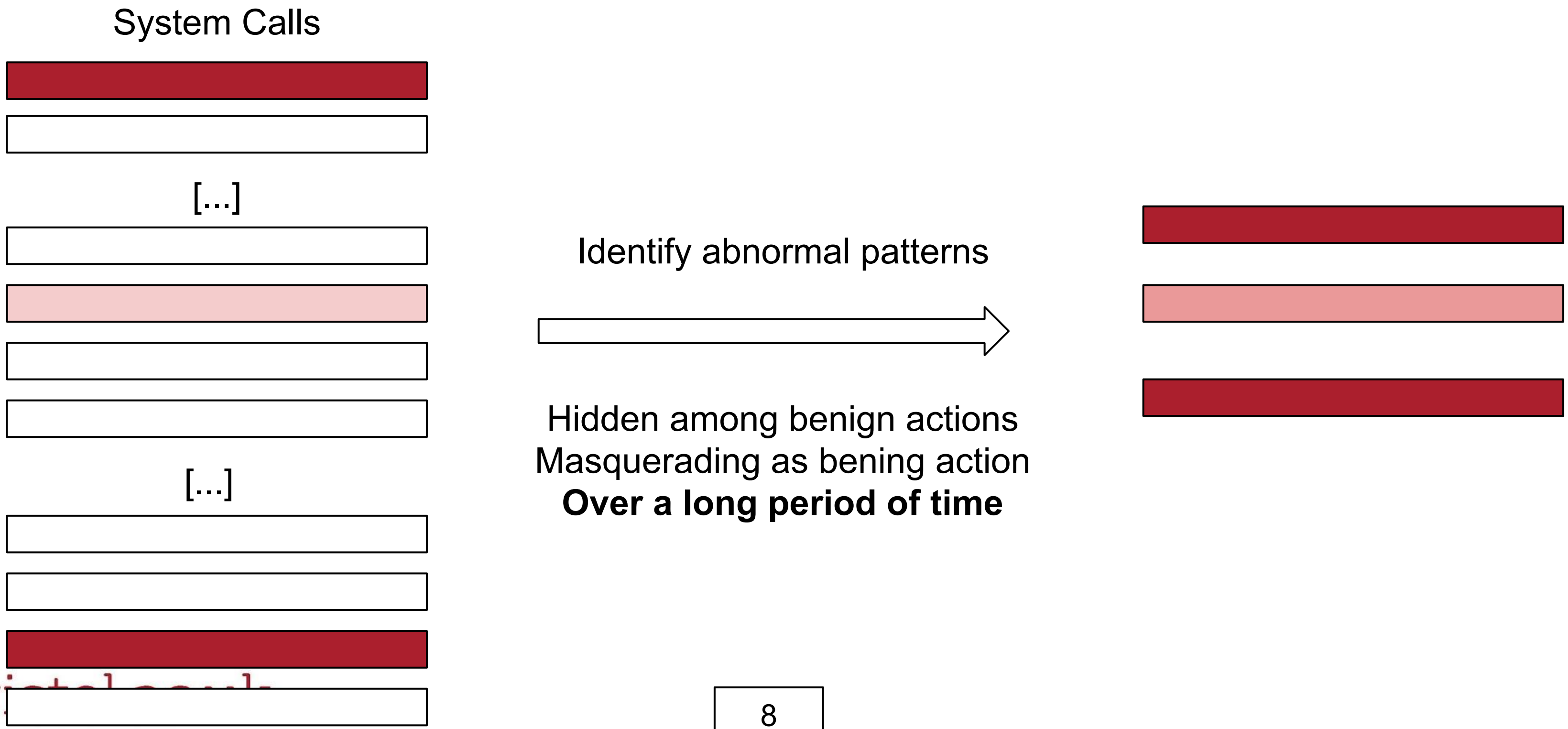


System call based intrusion detection

System call based intrusion detection

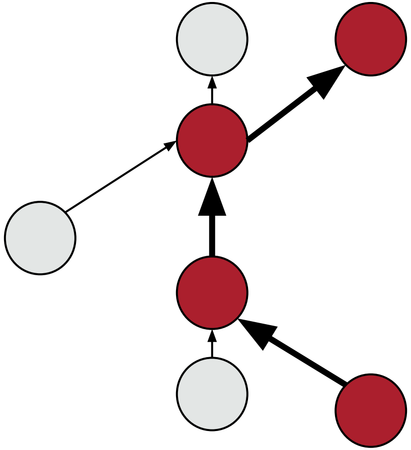


System call based intrusion detection



Provenance-based intrusion detection

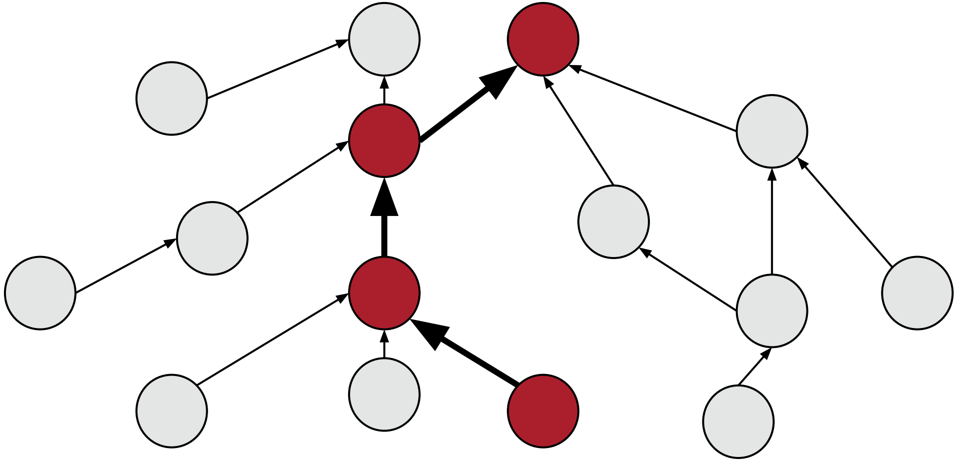
- **Intuition**: provenance graph **exposes causality relationships** between events



bristol.ac.uk

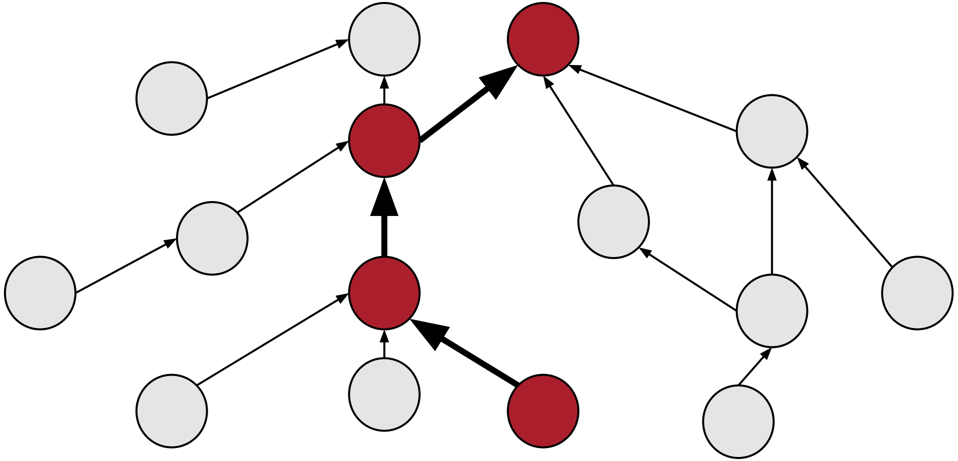
Provenance-based intrusion detection

- Intuition: provenance graph **exposes causality relationships** between events



Provenance-based intrusion detection

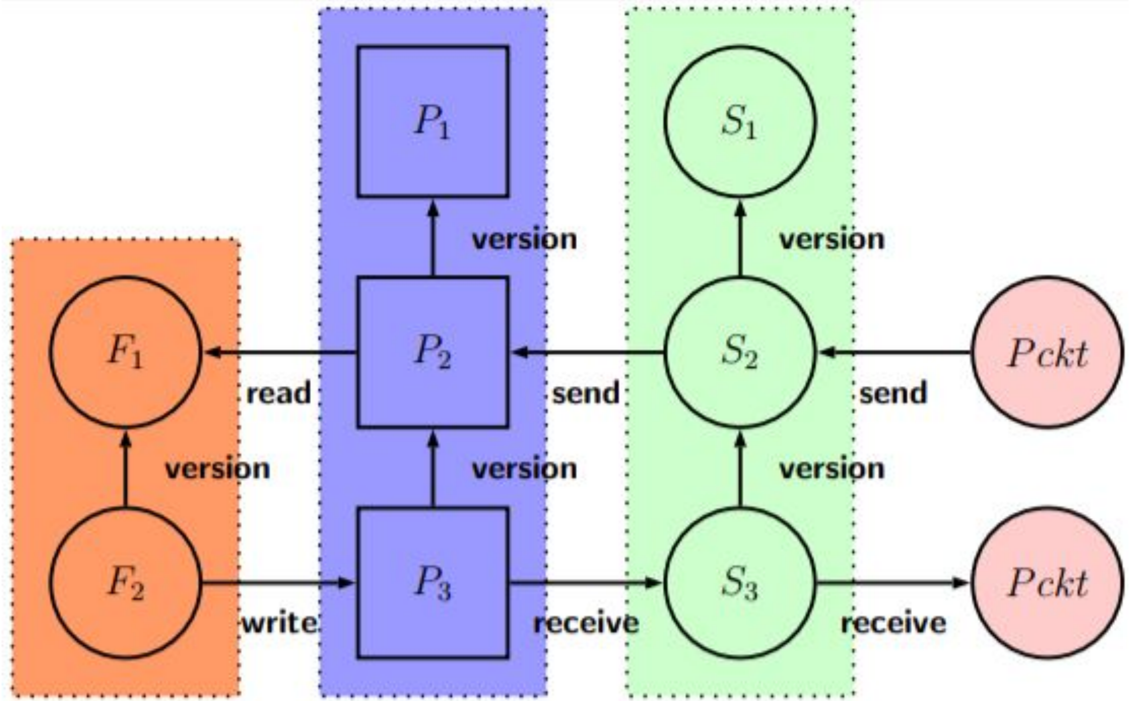
- Related system states are connected even across long period of time



What is provenance in an operating system?

- Represent interactions between system objects
- Represented as a **directed acyclic graph**
- Information Flows
- **Relationship** between **kernel object states**
- History of a system execution

Example provenance



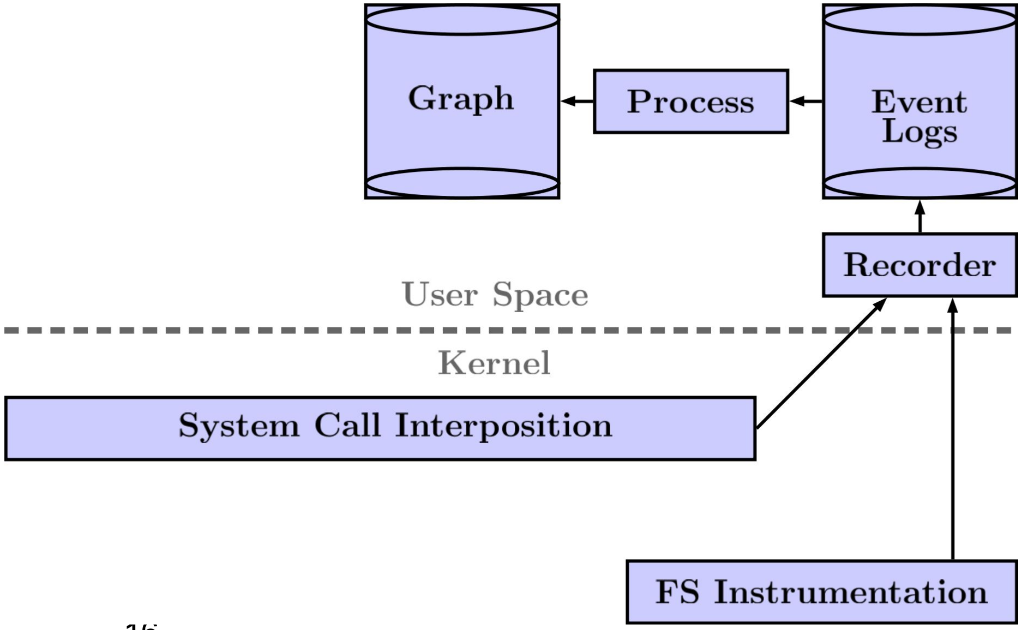
1. Capture
2. Processing
3. Detection

University of BRISTOL

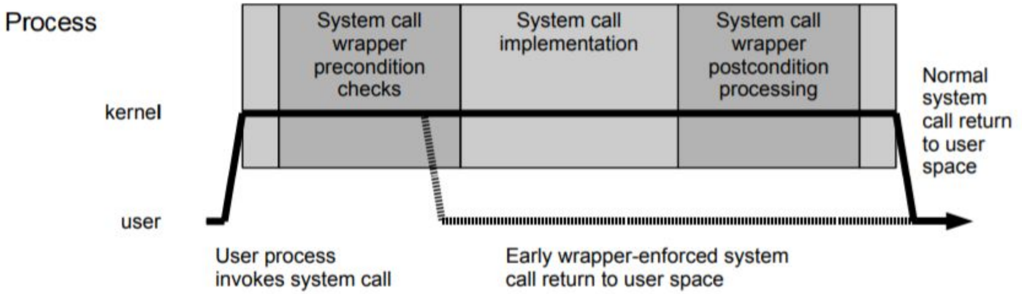
Provenance Capture

Provenance Capture Problem

- Muniswamy-Reddy et al. (USENIX ATC 2006)
- PASS
  - Untrustworthy
  - Hard to maintain
  - Project abandoned



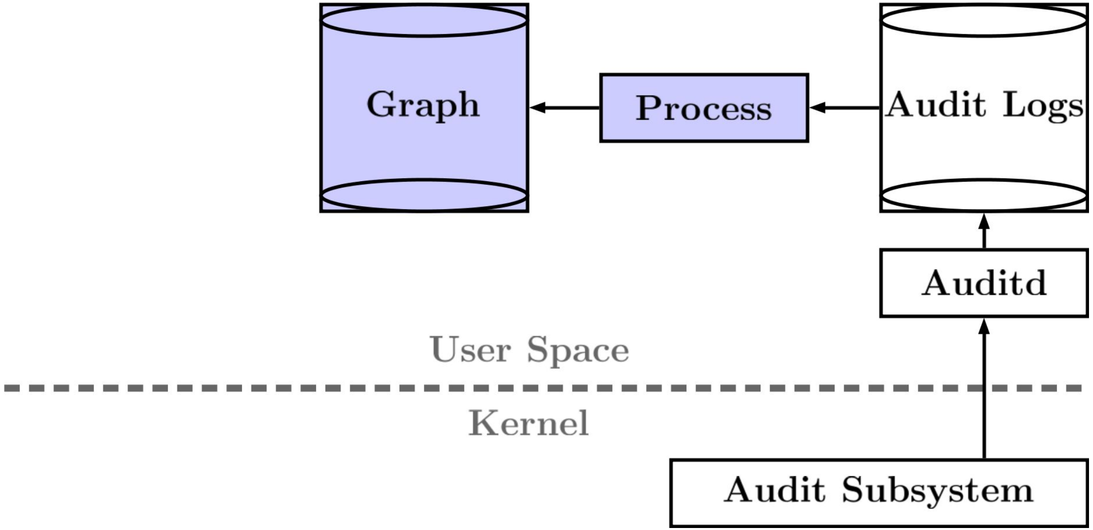
Interposition is un



- Watson WOOT 2007
- **Time-of-audit-to-time-of-use attack**
  - Race condition
- Syntactic Race
  - **different copy of parameters**
- Semantic Race
  - **Kernel state may change**

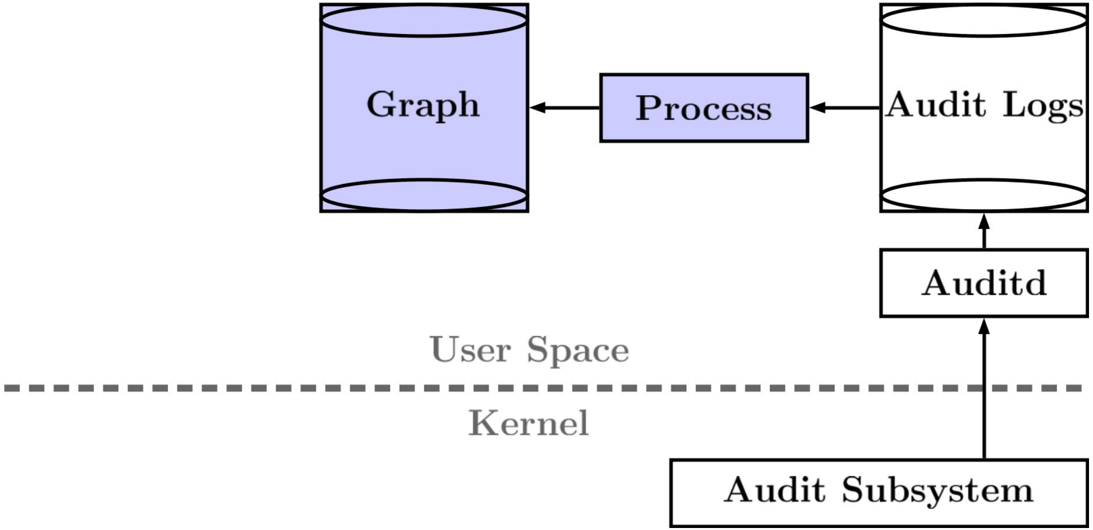
Provenance Capture Problem

- Gehani et al. ACM/IFIP/USENIX MW 2012
- SPADE
  - Easy to maintain
  - … did not quite work
    - ➢ Trustworthy, but
    - ➢ Not accurate



Provenance Capture Problem

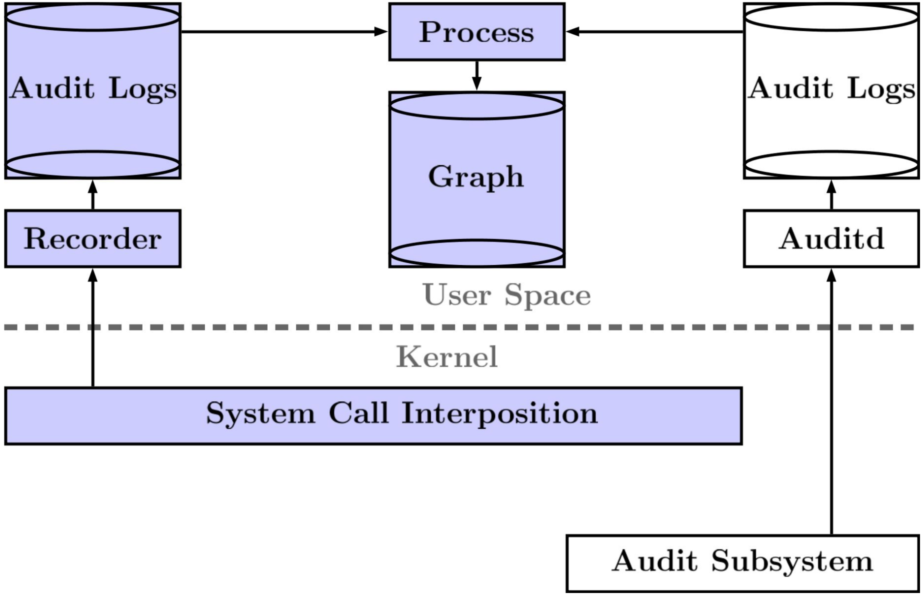
- Gehani et al. ACM/IFIP/USENIX MW 2012
- SPADE
  - Easy to maintain
  - … did not quite work
    - ➢ Trustworthy, but
    - ➢ Not accurate



- Missing kernel state
- Hard to infer causality

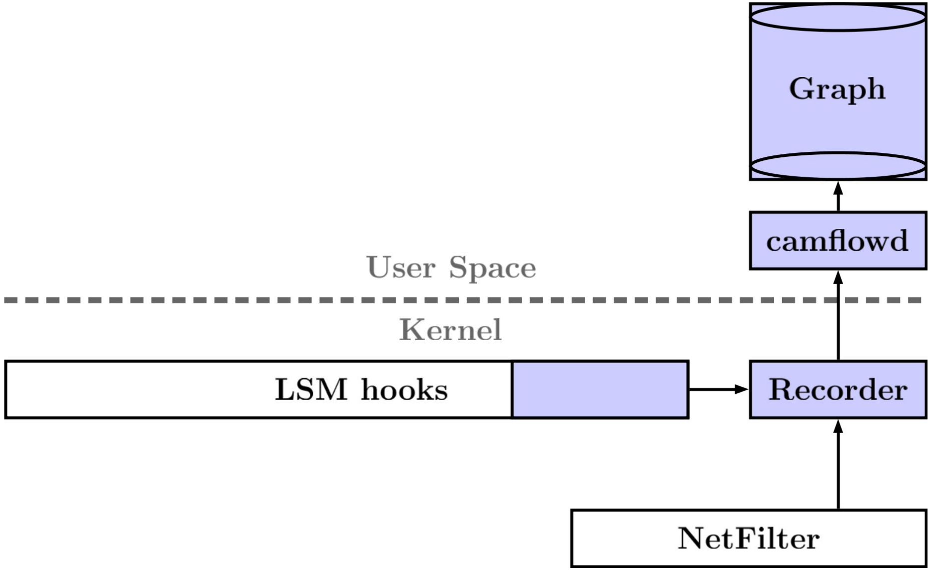
Provenance Capture Problem

- Gehani et al. unpublished
- SPADE v3
  - Back to square one
  - Same vulnerability
  - Hard to maintain



Fix: CamFlow

- Pasquier et al. IEEE TCC 2015 and ACM SoCC 2017
  - Rely on reference Monitor
  - **Trustworthy**
  - **Easy to maintain** small extension of the LSM framework.

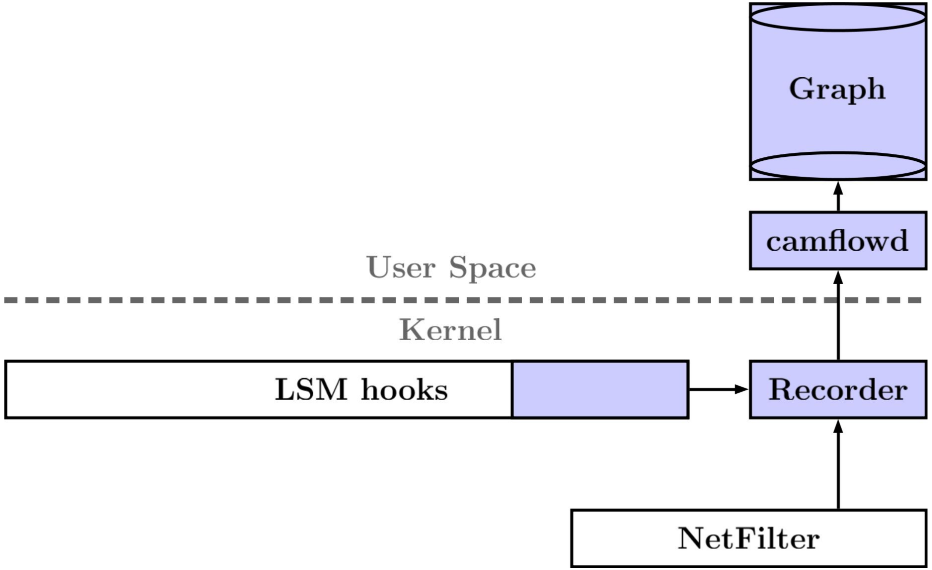


Fix: CamFlow

- Pasquier et al. IEEE TCC 2015 and ACM SoCC 2017
  - Rely on reference Monitor
  - **Trustworthy**
  - **Easy to maintain** small extension of the LSM framework.

2015-present
4.2.x to 4.20.x

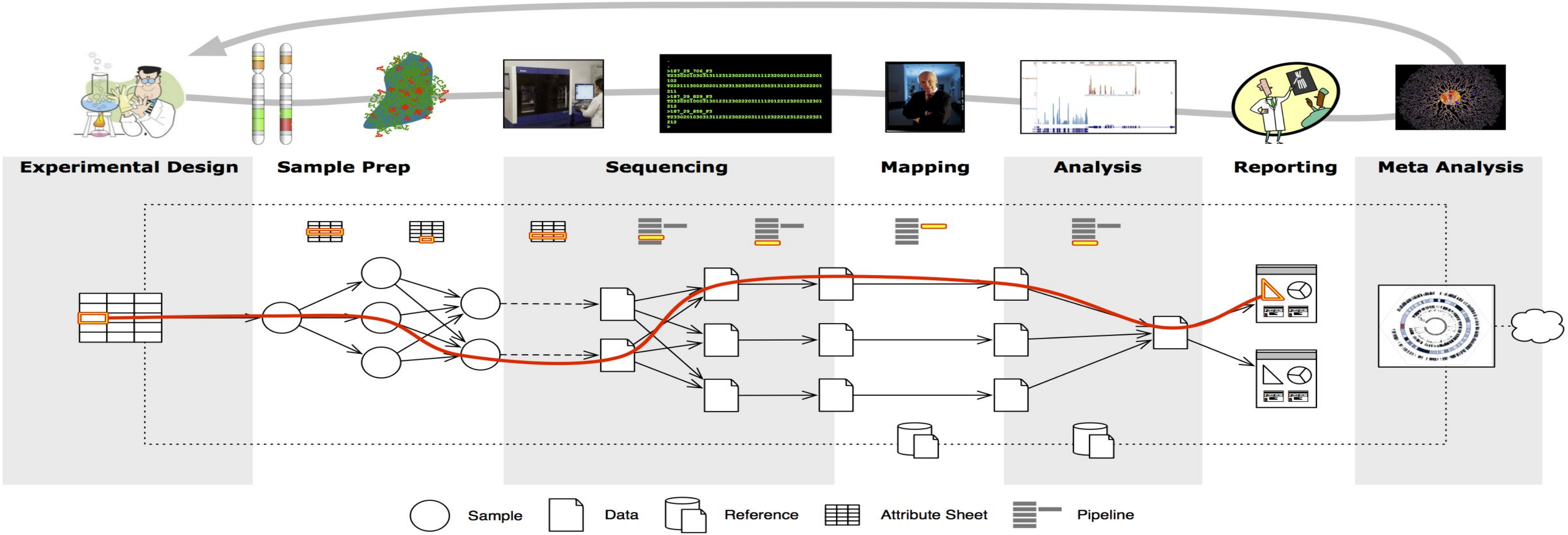
camflow.org



University of BRISTOL

Processing provenance

Provenance in science



bristol.ac.uk

Provenance-based security

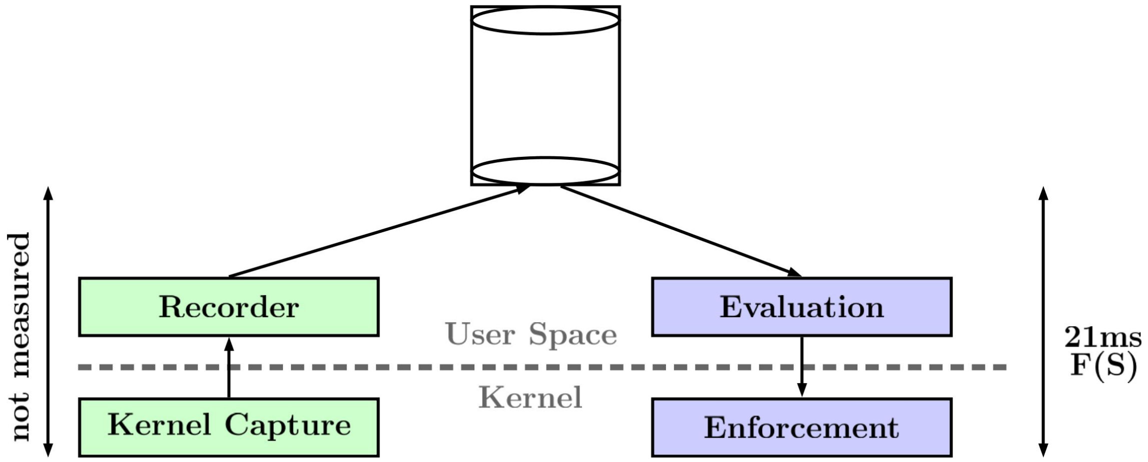
- Provenance-based access control
  - A provenance-based access control model, IEEE PST 2012
- Loss Prevention Scheme
  - *Trustworthy Whole-System Provenance for the Linux Kernel, USENIX Security 2015
- Intrusion Detection
  - FRAPpuccino: fault-detection through runtime analysis of provenance, USENIX HotCloud 2017
- Moving towards complex runtime graph analysis

Provenance-based security

- Provenance-based access control
  - A provenance-based access control model, IEEE PST 2012
- Loss Prevention Scheme
  - *Trustworthy Whole-System Provenance for the Linux Kernel, USENIX Security 2015
- Intrusion Detection
  - FRAPpuccino: fault-detection through runtime analysis of provenance, USENIX HotCloud 2017
- Moving towards complex runtime graph analysis
- *overhead is a function of total graph size, a graph which grows indefinitely
  - 21ms overhead per network packet, on small graphs

Policy evaluation problem

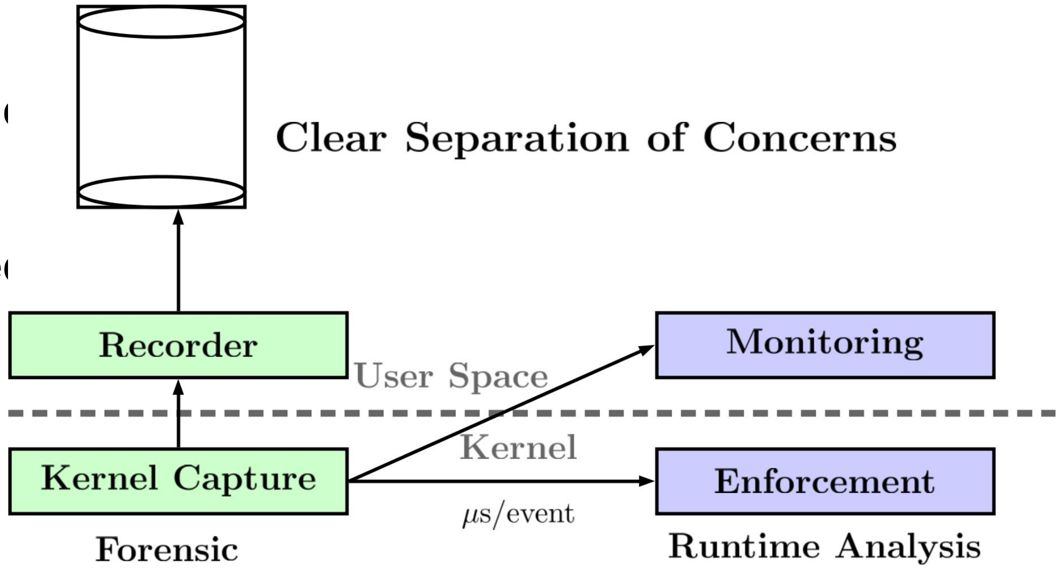
- Bates et al. USENIX Security 2015
  - Loss prevention scheme
  - Architecture not designed for enforcement
  - Very high latency
- Reduce graph size?
  - Pasquier et al. IEEE IC2E 2016
  - Bates et al. ACM TOIT 2017
- Does not quite save the problem



bristol.ac.uk

Fix

- Pasquier et al. to CCS 2018
  - Introduce separation of Concerns.
  - Can make separate trad[e] offs
  - Programmable loadable kernel provenance-base[d] policy module



Intrusion Detection

bristol.ac.uk

University of BRISTOL

Intrusion Detection

Work in progress, preliminary results

bristol.ac.uk

Provenance-based Intrusion Detection

- Flat logs are hard to analyse
  - Han et al. USENIX TAPP 2018
- Principle first introduced in Han et al. USENIX HotCloud 2017
  - First paper on the topic!
- We target cloud application
  - Relatively well defined behaviour
- Build a model of system behaviour
  - in a controlled environment
  - from a representative workload
- Detect deviation from the model
- Several approaches being explored…

Detecting intrusion

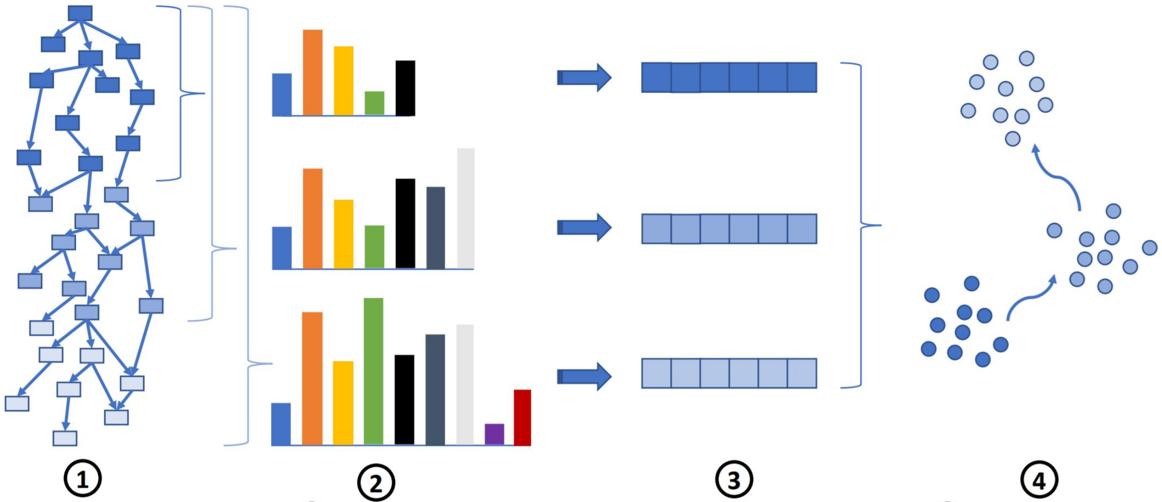


Fig. 1: UNICORN ① takes a streaming provenance graph, ② periodically summarizes graph features into histograms, and then ③ creates fixed-size graph sketches. The resulting clustering-based model ④ captures the dynamics of system execution. During deployment, graph sketches are created through the same steps (①, ② and ③) and then compared against the model in ④.

How well does it work?

Experiment	Precision	Recall	Accuracy	F-Score
StreamSpot	0.98/[0.48-1.0]	0.93	0.96/[0.50-0.82]	0.94
DARPA Cadets	1.0	1.0	1.0	1.0
wget Baseline	1.0	0.88	0.91	0.94
wget Interval	1.0	0.84	0.89	0.91

TABLE IV: Experimental results. We estimate StreamSpot's accuracy and average precision (in [red]) from the figure included in the paper, which does not report exact values. They did not report recall or F-score.

How well does it work?

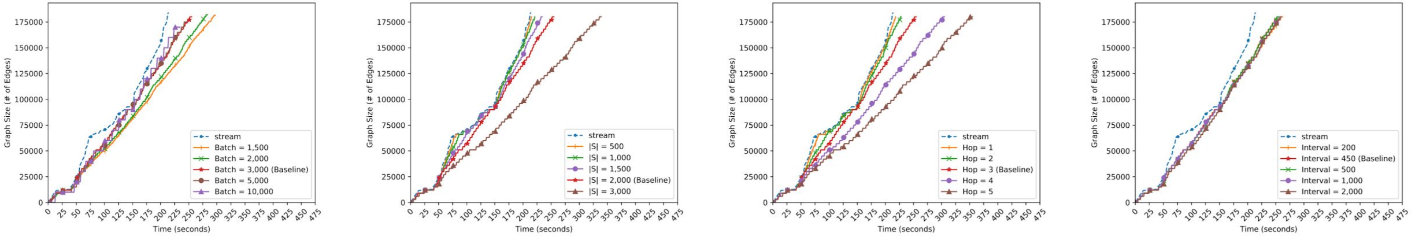


Fig. 3: Total number of processed edges over time (in seconds) of UNICORN in a `wget` experimental workload with varying batch sizes (left), sketch sizes (middle left), hop counts (middle right), and intervals of sketch generation (right). Dashed blue line represents the speed of graph edges streamed into UNICORN for analysis. Red baseline has the same configurations as those used in our experiment and indicates the values of the controlled parameters (that remain constant) in each figure.

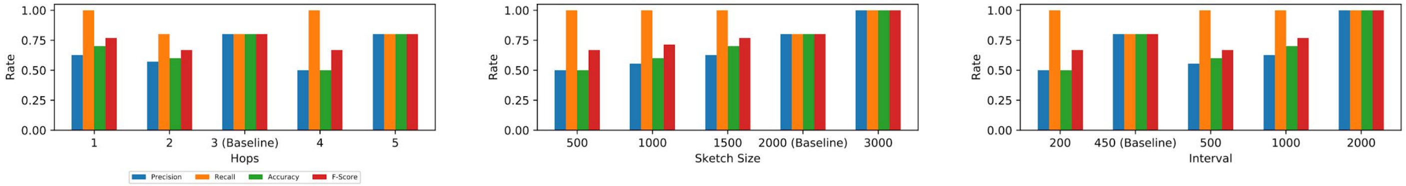


Fig. 4: Detection performance (precision, recall, accuracy, and F-score) with varying hop counts (left), sketch sizes (middle), and intervals of sketch generation (right). Baseline values are used by the controlled parameters (that remain constant) in each figure.

bristol.ac.uk

Some insights

- We can detect intrusion out of graph structure with little metadata
  - Vertex type (thread, file, socket etc…)
  - Edge type (read, write, connect etc…)
- Processing speed
  - Current prototype
  - Data generation speed < processing speed!

Future direction

Research Trajectory 1/2

- Doing proper evaluation is hard!
- Dataset are hard to generate
    - What is a good quality dataset?
- Hard to compare across papers, a lot is not available
    - Experiments (i.e. attacks)
    - Capture Mechanisms
    - Analysis pipelines
- Leads to unsatisfactory evaluation
    - I may be able to compare to similar techniques (may reuse dataset)
    - … very hard for unrelated one

Research Trajectory 2/2

- Extending to distributed systems and IoT
    - An auditable IoT environment?
    - EPSRC DataBox project (verifiable ledger)
- Solving the many provenance challenges
    - Storage (Database)
    - Trust (Crypto/Hardware)
    - Representation (HCI)

University of BRISTOL

Thank you, questions?

tfjmp.org

camflow.org

bristol.ac.uk